The CNVineta Tutorial for the manuscript:

CNVineta: A data mining tool for large case-control copy number variation data sets.

Wittig[1] M, Helbig[2] I, Schreiber S[1], Franke A[1]

[1] Institute of Clinical Molecular Biology, Christian-Albrechts-University, Schittenhelmstr. 12, 24105 Kiel, Germany

[2] Department of Neuropediatrics, University Medical Center Schleswig-Holstein, Kiel Campus, Arnold-Heller-Strasse 3, Building 9, 24105 Kiel, Germany

Contents

Introduction 1
A sample session using the provided HapMap data set 2
Visualizing a specific region (*IRGM* in YRI and CEU) 4
CNV burden 9
Genome-wide identification of rare CNVs 11
Genome-wide common CNV association analysis 12
Generating CNVineta input files 14
CNVineta input file format description 14
Converting CNV calling results to CNVineta input file format 16
An example conversion process 17

Introduction

CNVineta is a flexible data mining tool for the analysis of copy number variations (CNVs) in large case-control SNP array data sets. The tool is available as an R statistical package. CNVineta offers a flexible and fast access to CNVs by allowing for a quick graphical overview in large case-control datasets. In addition, CNVineta provides rapid access to the $\log_2$ of raw data ratios (LRR) and B-allele frequencies (BAF) of specific or all samples, thereby allowing for a fast verification of the underlying raw data. CNVineta is also equipped with analysis methods for genome-wide screening for associated rare as well as common CNVs. Hence, CNVineta is a unique data mining tool to rapidly explore CNVs in large case-control data sets.A sample session with CNVineta

The next pages guide the user through a sample session which uses the free available

HapMap data set [1]. The sample session starts with CNVineta initialisation followed by sample filtering, association screening for rare and common CNVs. Finally downstream verification of the predicted associations is included as well. Download the Affymetrix® 6.0 HapMap data set ready for CNVineta from http://www.ikmb.uni-kiel.de/cnvineta/. Unpack the splitted zip or tar.gz archives and remember the path where the unpacked HapMap data is on your computer. For example, if files are unpacked at "/foo/bar/" (e.g. Windows "E:\test\") remember the directory "/foo/bar/CNVineta/" (e.g. Windows "E:\test\CNVineta\"). If the 2.7 GB HapMap CNV dataset should not be downloaded, you may use the reduced data which is provided with the package. Please read the first paragraphs of the sample session until the discussion of the *filterSamples* command, which will explain the handling of the reduced dataset.
To start your first CNVineta session, download the CNVineta package from http://www.ikmb.uni-kiel.de/cnvineta/ and install the package in your R framework.

A sample session using the provided HapMap data set

First use the generic R command setwd() and set the working directory to the folder, which contains the HapMap data. After this command, all further R commands in this sample session are executable by copy and paste. Remember the directory where you have unpacked the HapMap data and execute setwd().

```
setwd("/foo/bar/CNVineta/")
```

If you want to go through the tutorial without copying and pasting the commands step by step you can run a script which does the entire CNV screening process. To do so please paste the following command:

```
source("example_cmds.R")
```

To perform the automatic screening by using the example dataset provided with the package, please execute

```
library(CNVineta); runCNVinetaExample("/my/output/is/here/")
```

Now you can start with the step-by-step tutorial. After installation, the CNVineta package is loaded with

```
library(CNVineta)
```

For the following analysis steps, a CNVineta object. has to be generated. If the reduced dataset is used, which is provided with the package, initialisation is slightly different and the following three commands can be skipped (continue after *samples2batches*).

```
objCNVineta = new("CNVineta")
```

This object, called objCNVineta in this tutorial, will combine information on the input files (binary.dat file, SNP array annotation file, sample annotation file, predicted CNVs file and RefGene file). Navigate to the directory with the sample files (as described above 'setwd()') or use full pathnames instead of filenames only and initialize the CNVineta object using

initializeCNVineta. This might take a minute or two, depending on the SNP array and sample number.

```
objCNVineta = initializeCNVineta(objCNVineta,
 snp.anno = "CNVineta.HapMap.Affy6.0_binary.dat.snp.idx.txt",
 ind.anno ="CNVineta.HapMap.Affy6.0_binary.dat.sample.idx.txt",
 binary.dat = "CNVineta.HapMap.Affy6.0_binary.dat",
 segments.orig = "CNVineta.HapMap.Affy6.0_binary.dat.segments.txt",
 refgene.anno = "refGene.txt")
```

For the subsequent analysis, different samples may have to be assigned to new or other batches. Information on batches, which will also be used for case-control analysis later on, has to be provided in a tab delimited file, which has two columns with headers "batch" and "sample". Type "?samples2batches" in R for an example (question mark followed by a command name opens a help page).

```
objCNVineta = samples2batches(objCNVineta, "samples2batches.txt")
```

For the reduced package data set please start with the following command and set a path with setwd() to the location where the output files should be exported to.

```
objCNVineta = initializeCNVinetaExample()
setwd("/my/output/is/here/")
```

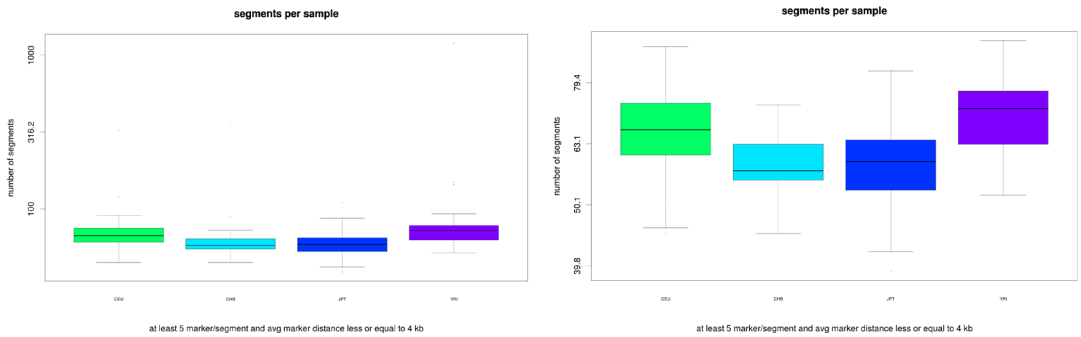
SNP array data analyzed for CNVs usually shows a wide variation of copy numbers per sample. CNVineta can identify samples with excessive copy number segments per sample. Implemented filter criteria are either "w" for discarding samples with boxplots whiskers as threshold. Or "q" with the quantiles as threshold.

The following command will generate a boxplot with the number of segments per sample in each batch before and after filtering as well as a summary table including information on the number of samples included in each batch before and after filtering.

```
objCNVineta = filterSamples(objCNVineta, out.file="filter")
```

```
[1] "after filtering segments per sample:"
  batch samples.original samples.filtered delta delta.percent
1 CEU                 90               88     2           2.2
2 CHB                 45               43     2           4.4
3 JPT                 45               43     2           4.4
4 YRI                 90               87     3           3.3
```

In this analysis, 2 out of 90 CEU samples were discarded. Two graphs will appear in the working directory (use getwd() to determine the actual working directory), which show batch-wise boxplots before (filter01.jpg) and after (filter02.jpg) filtering.



After this preliminary quality control, the data is ready for visualization and further analysis. Please be aware to apply well-established sample filters before you work with CNVineta, i.e. identification of duplicate or related individuals as well as "ethnic outliers" should be performed prior to subjecting your data to CNVineta. Also, the CNV prediction tool specific filters such as MAPD and contrastQC filter for Affymetrix® Power Tools analyses or mean LRR and mean BAF scatterplots for QuantiSNP analyses should be applied.

Visualizing a specific region (*IRGM* in YRI and CEU)

A specific chromosomal region can be visualized using the *visualizeRegion* function. This function is an essential function within the CNVineta package, as other functions such as the initial visualisation step for genome-wide association screening use this function. For the HapMap example data set, we will use a deletion polymorphism upstream of *IRGM*, which has been shown to be associated with Crohn's disease [2]. This analysis will produce a table with the summary of CNVs found in a specific chromosomal region and a *regional overview plot*. In this example, case status is assigned to YRI (upper panel in Vineta plot) and control status to CEU (lower panel).

visualizeRegion(CNVdata = objCNVineta, batch.case=c("YRI"),batch.control=c("CEU"), chrom = "5", start = 150100000, end = 150300000)

```
261  samples to process
26  samples with CNV
[1] "cases   : 26 segments in  26 samples"
[1] "controls: 0 segments in  0 samples"
[1] "got 7 transcripts"
    size  0 1  2 3 4+ #del samples #dup samples #CNP samples
YRI   87 24 2 NA 0  0           26            0           26
sum   87 24 2 NA 0  0           26            0           26
    size 0 1  2 3 4+ #del samples #dup samples #CNP samples
CEU   88 0 0 NA 0  0            0            0            0
sum   88 0 0 NA 0  0            0            0            0
                          Deletion Duplication Copy Number Polymorphism
Pearson's Chi-squared 9.034387e-08           0             9.034387e-08
Fisher's Exact        1.406624e-09           1             1.406624e-09
[1] 150100000 150300000
Warning message:
In chisq.test(chi_table) : Chi-squared approximation may be incorrect
```

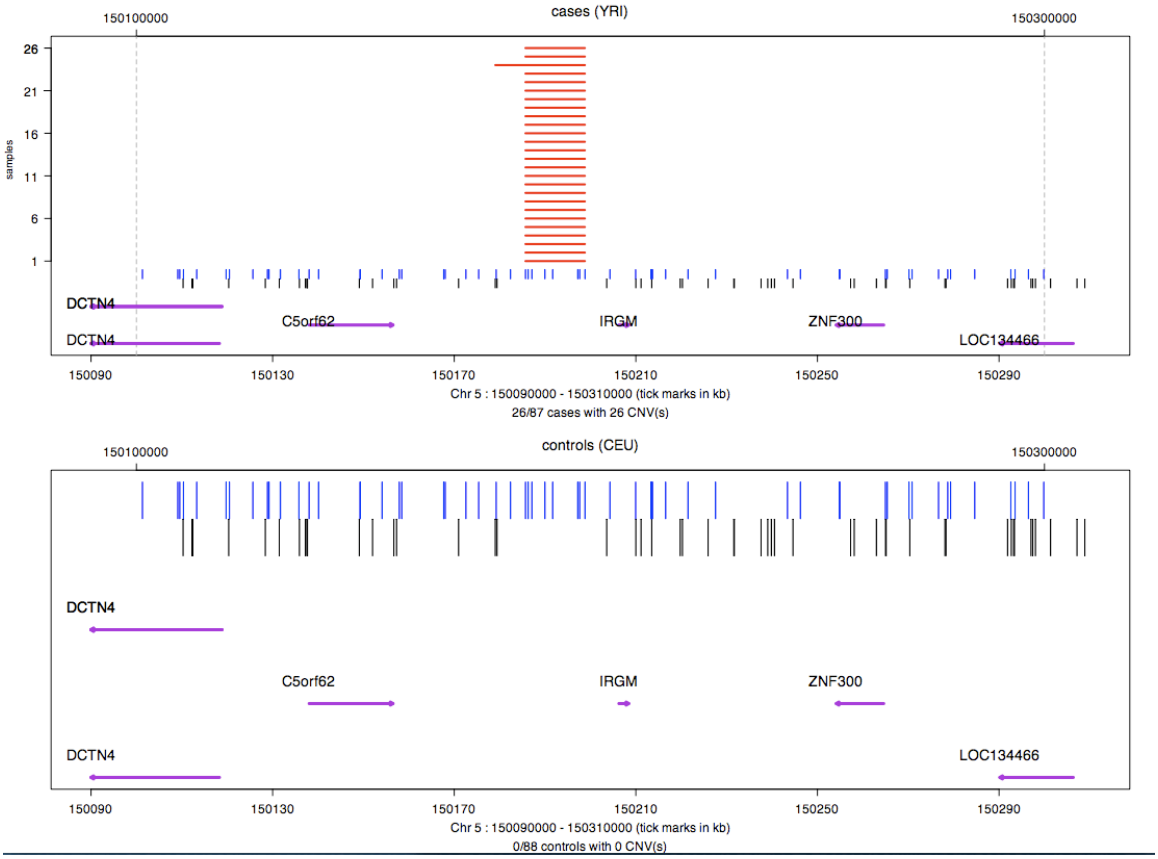
In summary, this analysis detects a strong association of a deletion assigned to the case cohort (YRI). In fact, 26 out of 87 individuals of this cohort have a predicted deletion, while

none out of 88 individuals in the CEU group has a deletion.

Just be aware that in this example, the association analysis is performed over the entire region on chromosome 5 (*start = 150100000, end = 150300000)* and the association analysis does not take into account the location of the deletions or whether there is overlap between CNVs. It is a fundamental problem for any CNV association analysis that p-values do not provide information on the relative location of the segments. We will provide a solution with the later described functions *dox0* and *do.log.regression*.

In the above example, R produces a warning concerning the Pearson's Chi-squared test. This is to be expected since the number of deletions in the control cohort is zero. Ignore the warning for now, as the *visualizeRegion* function also provides a Fisher's exact test result.

For a first orientation, the *visualizeRegion* function also generates a regional overview plot. This plot stacks all individuals with copy number variations in cases and controls and visualizes the extent of the copy number variation relative to genes. Black ticks correspond to SNPs, blue ticks indicate non-polymorphic probes. Deletions are marked in red, while duplications are in blue. By plotting all individuals with CNVs in cases and controls in a given chromosomal region in a single plot, CNVineta provides a quick overview of the segment data in the entire sample set that is used for the association analysis.



As can be seen in the regional overview plot, predicted deletions upstream of *IRGM* are only

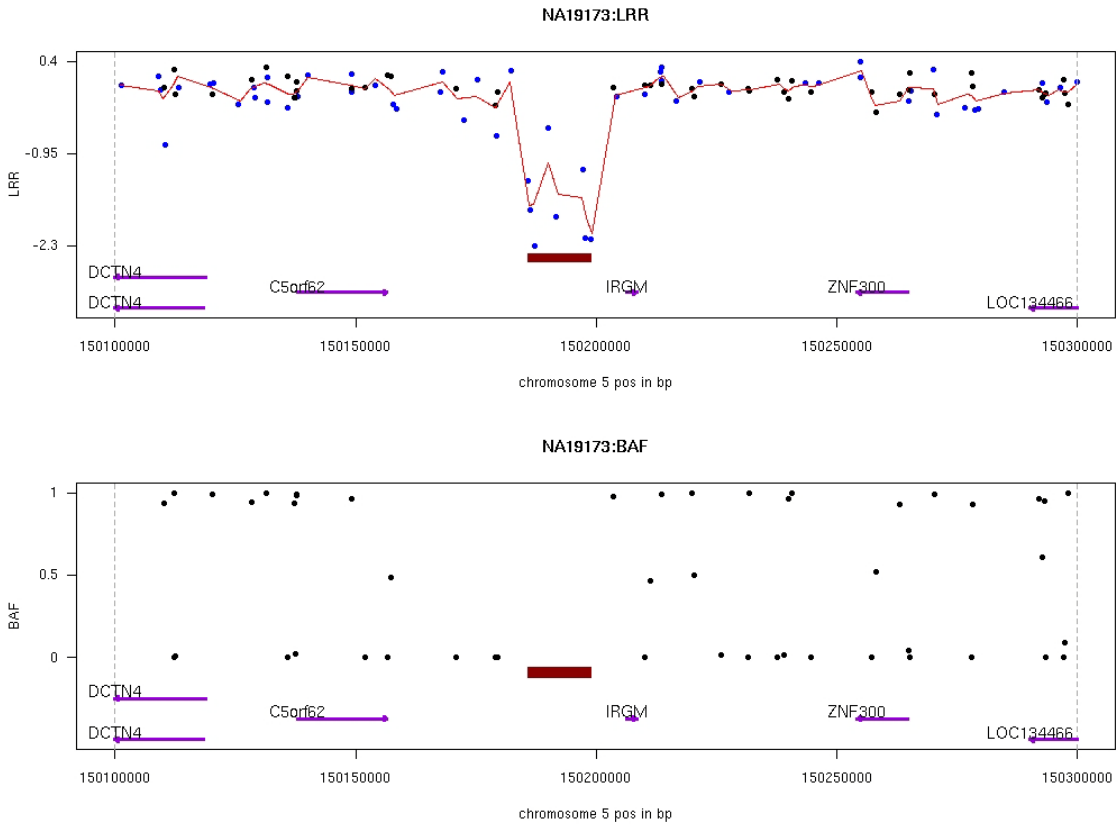
present in YRI, while the deletion seems to be absent in CEU. In addition, the deletion is only covered by non-polymorphic probes on the Affymetrix SNP 6.0 array (blue ticks), while no SNP assays are present in this region. The *visualizeRegion* function also includes triggers that will output the table and sample names into a file.

However, before further analysis is performed, the LRR of the deletion in all samples should be investigated. While no SNPs cover this CNV, the BAF plot is less important for this example but generated as well. The deletion polymorphism is only covered by few non-polymorphic probes and might possibly be a false positive CNV due to incorrect calls by the CNV calling algorithm.

The *visualizeRegion* function has several triggers, which allow for a look at the raw data. Setting the out.file trigger, as well provides graphs in separate files.

```
visualizeRegion(CNV data =
objCNV ineta,batch.case=c("YRI"),batch.control=c("CEU"),chrom = "5" ,
start = 150100000, end = 150300000, raw.data.plots=TRUE,
out.file="IRGM_overview", print.tables=TRUE)
```

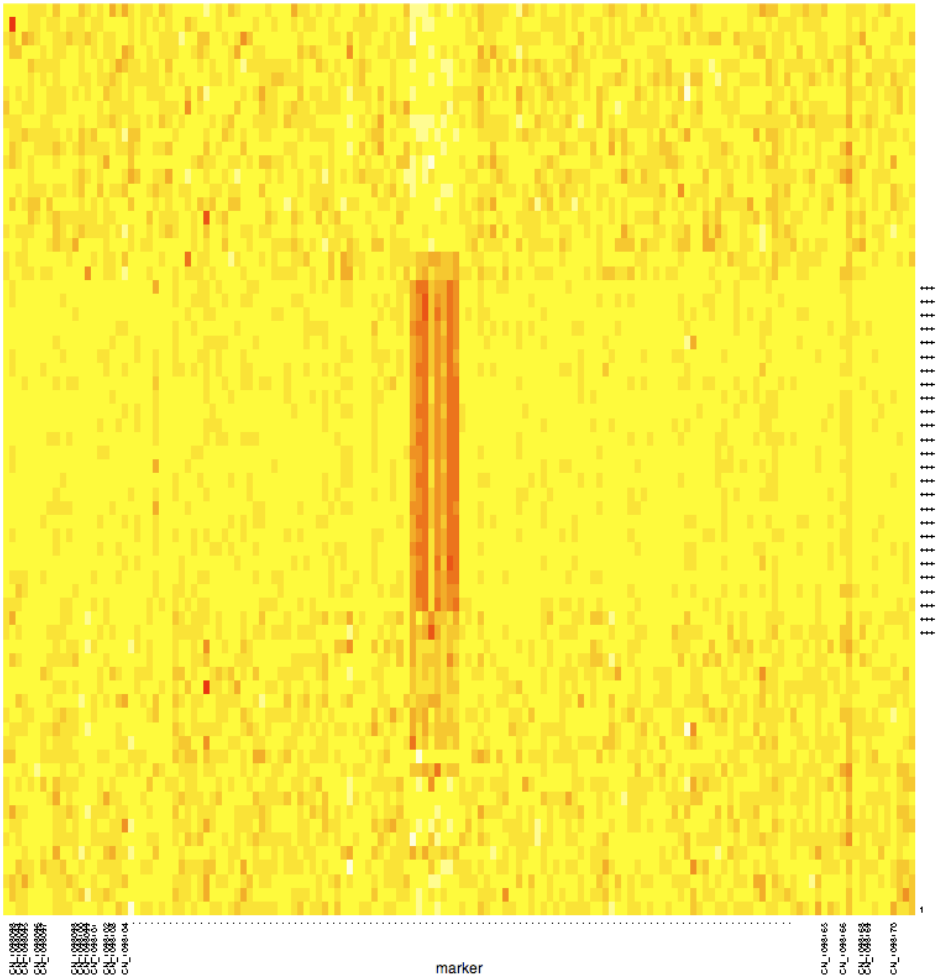
Look at all files in the working directory starting with "IRGM_overview". This will include .jpeg files for the regional overview plot, raw data LRR and BAF plots of every sample that has a predicted CNV in this region and a table which annotates all predicted CNVs in cases and controls for that region. Below is an example plot for an individual of the YRI cohort.



The upper panel shows the LRR, the lower panel shows the BAF in individual NA19173. The red line indicates the average LRR value, the red block demarcates the deletion predicted by the algorithm. As can be seen, the deletion predicted by the CN algorithm is accompanied by a shift in the LRR of several neighboring probes, therefore suggesting that the deletion is actually real and not an artifact of the CNV calling algorithm. When SNP assays cover a deletion you expect either a loss of heterozygosity (heterozygous deletion) or total loss of the BAF signal (homozygous deletion). For duplications, the BAF should spread wider.

The raw data plot inspection as described above is a good way to identify false positives. But what about false negatives? What if the CNV calling algorithms failed to identify CNVs which are present in some samples, but not in others. By using the *heatmap* command, the samples with CNVs are compared to random samples in the cohort that do not have predicted CNVs in this region.

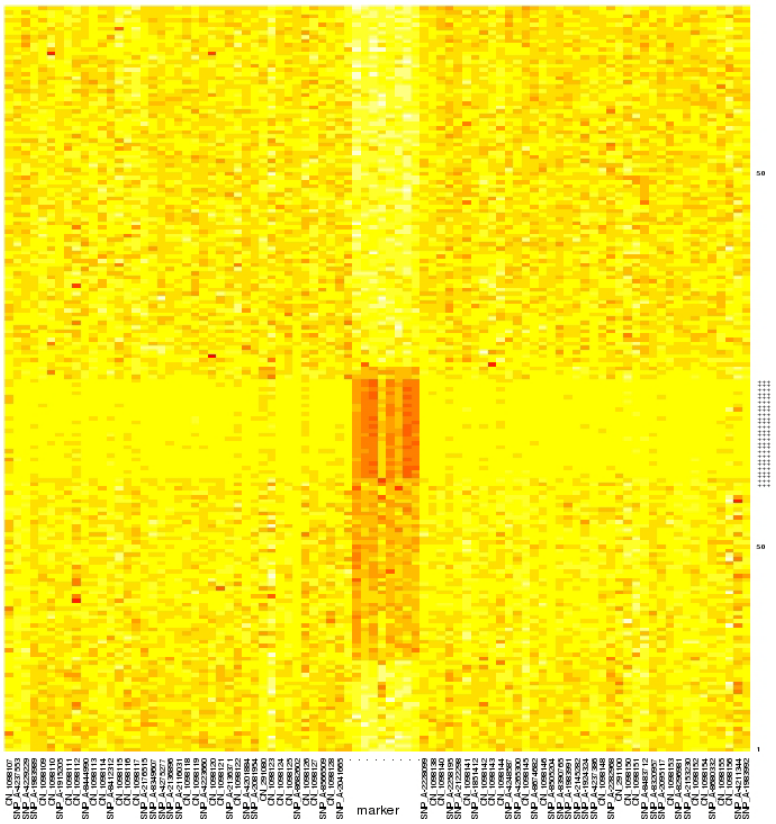
```
visualizeRegion(CNVdata =
 objCNVineta,batch.case=c("YRI"),batch.control=c("CEU"), chrom = "5" ,
 start = 150100000, end = 150300000,add.heatmap=TRUE)
```



In this heatmap, case samples with predicted deletions are indicated by three "+" symbols, control samples with predicted CNV are indicated by one "-" symbol (not present in above dataset). Changes in the LRR are indicated by color changes (red = lower LRR). As can be seen, there is a strong difference between individuals with deletions and individuals without deletions. However, there are several individuals, which appear to have a deletion. These deletions, however, were not called by the CNV calling algorithm, indicating that some samples might have CNVs that were not identified by the CNV calling algorithm. Therefore, checking the entire cohort in a cohort-wide heatmap should be the next step.

A heatmap for the entire sample set helps to identify the number of false negatives. A heatmap sorting function is also implemented, in order to trim the query to cover only the CNV region in question. This generates a heatmap, which has better sorted rows. The trigger *flanking.snp* defines the number of SNPs that should be added left and right to the requested region in the heatmap.

```
getHeatmap(CNVdata= objCNVineta,batch.case = c("YRI"),
 batch.control = c("CEU"), chrom = "5" ,start = 150180253,
 end = 150200253, flanking.snps=40, cohort.wide=TRUE)
```



The implemented sorting functionality can even be more helpful to identify samples with LRR shift but that were not called as CNVs. Sorting is based on the difference between the mean LRR of the requested region and the mean LRR of its flanks. The probe sets of the requested region are labeled with dots. Trimming of regions helps achieve the best results.

In summary, using CNVineta, we have shown that deletions upstream of *IRGM* are significantly more frequent in YRI compared to CEU and that most YRI individuals have homozygous deletions. Some deletions, which seem to be heterozygous, seem to be missed by the segmentation algorithm and should be examined further.

CNV burden

Calculating the CNV burden/load of a sample set is increasingly performed throughout the literature. In order to perform a CNV burden/load analysis, a CNV singleton screen [6] and CNV load statistics are included. The singleton screen scans for CNVs, which are sample specific and do not overlap with other CNVs. This screen is either specific for deletions, or duplications or can be used for all copy number states (deletions and duplications).

CNV.singleton(CNVdata = objCNVineta, batch.case=c("YRI"),batch.control=c("CEU"), visualize = FALSE)

The result are provided in three data tables which list the number of samples, number of singletons, singletons per sample, number of samples with singletons and number of

singletons overlapping genes for the three different CNV singleton types.

```
[1] "singletons:"
         sample.number singleton.number singletons.per.sample
cases               87              309              3.551724
controls            88              205              2.329545
         samples.with.singletons singletons.overlapping.genes
cases                         82                           122
controls                      75                            98
[1] "del.singletons:"
         sample.number singleton.number singletons.per.sample
cases               87              291              3.344828
controls            88              164              1.863636
         samples.with.singletons singletons.overlapping.genes
cases                         77                           102
controls                      70                            71
[1] "dup.singletons:"
         sample.number singleton.number singletons.per.sample
cases               87              294              3.379310
controls            88              139              1.579545
         samples.with.singletons singletons.overlapping.genes
cases                         51                           238
controls                      52                            93
```

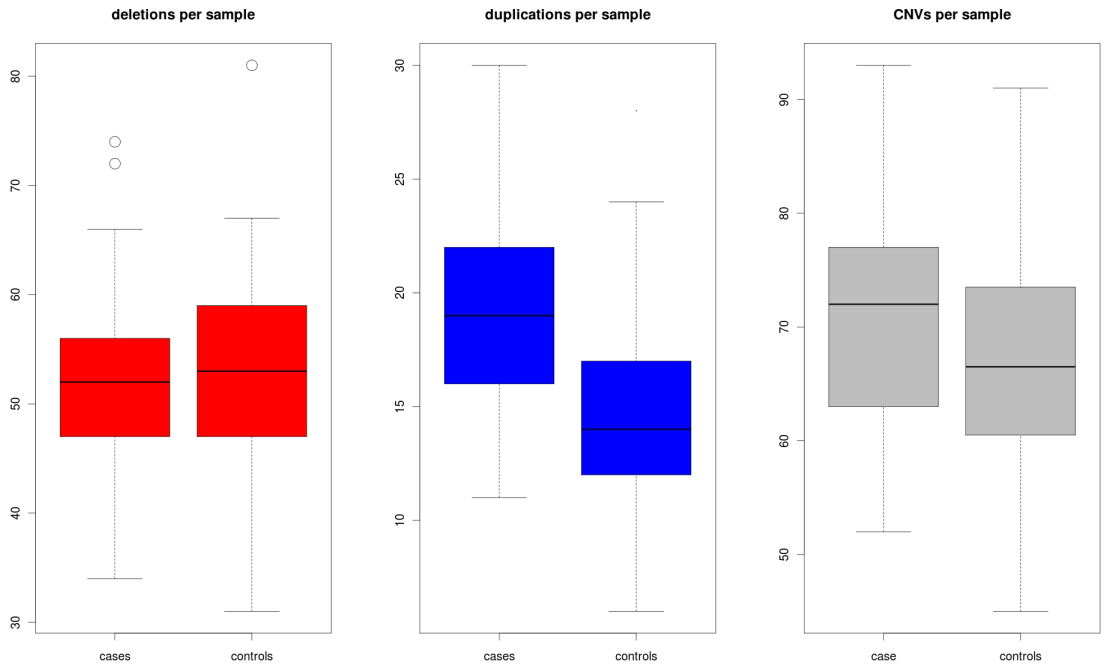
If the parameter *visualize* is set to TRUE, regional overview plots are generated automatically. We suggest also setting the parameter out.file, because a large number of singletons can be expected per analysis.

The CNV load functionality, assessing the number of segments per sample, is already included in the filtering process where filtering is based on the number of segments per samples. To combine different batches to cases and/or controls and calculate the case-control specific CNV load, you can use the *CNV.load* function.

```
CNV.load(CNV.data = objCNV.ineta, batch.case=c("YRI"),batch.control=c("CEU"))
```

The function produces following output:

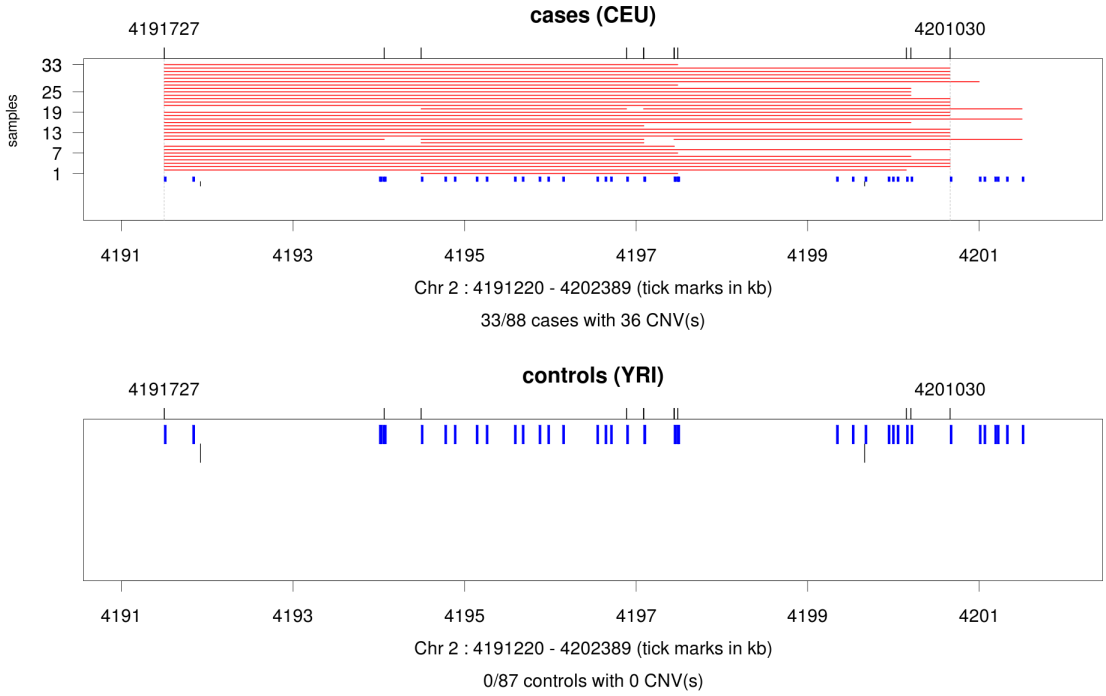
```
         samples median(deletion) median(duplication) median(CNV)
cases         87               52                  19        72.0
controls      88               53                  14        66.5
```



Genome-wide identification of rare CNVs

CNVineta offers the possibility to identify CNVs, which are only found in cases, but are absent in controls. We refer to this analysis as X0-analysis ("x deletions or duplications in cases vs. 0 in controls") and it is invoked by the *dox0* command. The basic idea of the genome-wide CNV analysis is reducing the genomic positions for analysis to a minimum. CNVineta defines so-called "atoms" (Supplementary Figure S2), which represent marker positions where a predicted CNV in any sample either starts or ends. These atoms are then used for the entire sample set to create a atomic matrix for the genome-wide scan. The x0-analysis simply looks for atoms, which have CNVs in cases but not in controls (Supplementary Figure S4). The graph below shows a deletion on chromosome 1 which is only found in YRI, but not in CEU. A complete table with these regions can be generated using specific triggers for *dox0*.

```
dox0(CNVdata = objCNVineta, batch.case=c("CEU"),batch.control=c("YRI"),
 draw=TRUE, min.diff=5, max.one.side=1, cases.more.affected=TRUE, out.file="x0")
```



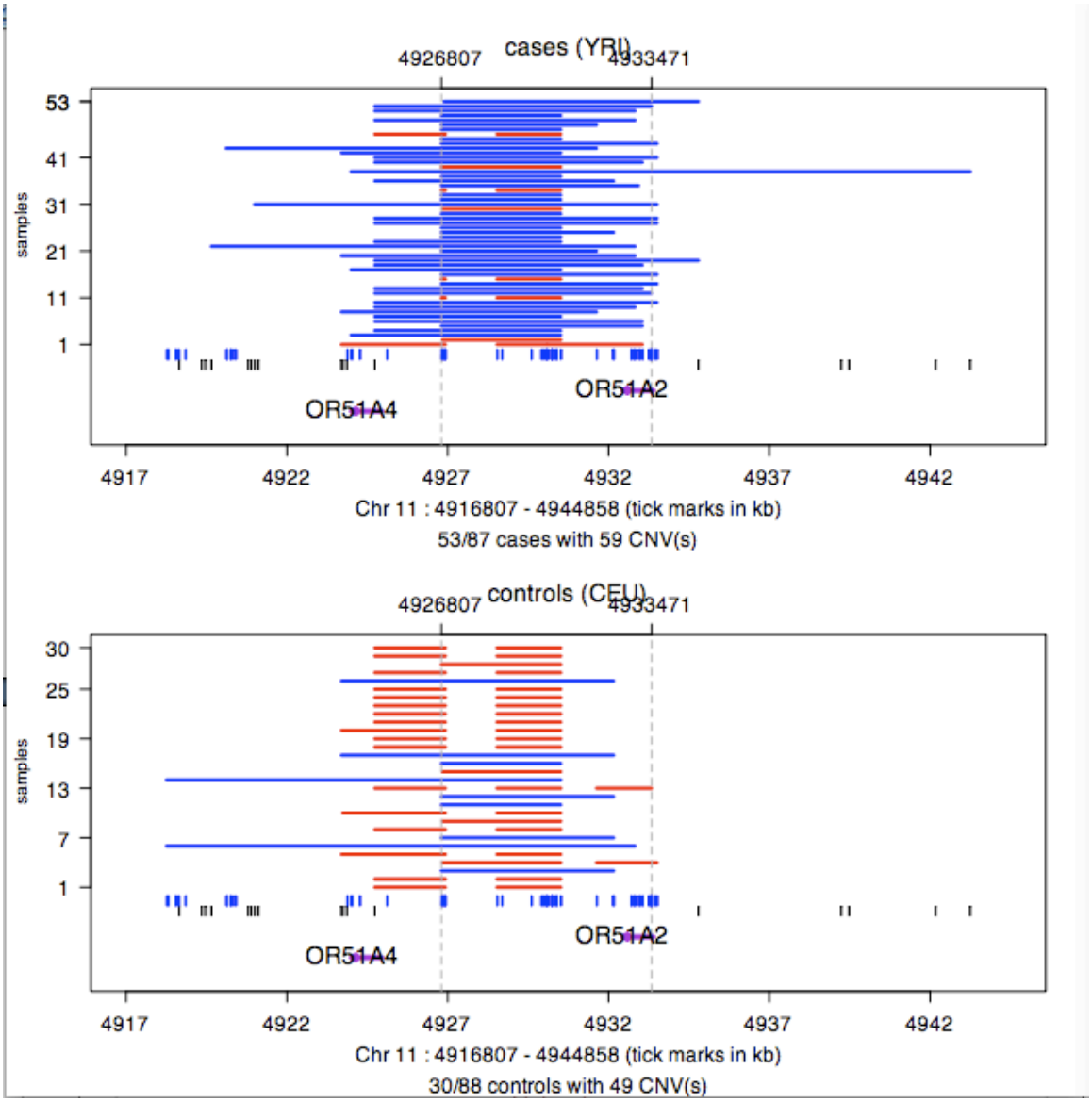
The command above generated a large number of image files starting with "x0" in your work directory. When screening these graphs, each tick at the upper x-axis of each graph marks an atom, which adheres to the specified x0 rule.

Genome-wide common CNV association analysis

The genome-wide CNV analysis function of CNVineta works on the same atomic matrix as the x0-analysis. However, the analysis is conceptually different. The genome-wide CNV analysis invoked by *do.log.regression* initiates a genome-wide association study on the elements of the atomic matrix (Supplementary Figures 5 and 6). When adjacent atoms or atoms within a defined window size (trigger *peak.window*) show significant p-values, these atoms are merged to one peak. This reduces the amount of target regions that have to be visually screened but does not minimize the information. A peak table is returned by *do.log.regression* by default (same as for *dox0*). This peak definition is only used for visualization of the data.
In the following example, you will see a significant locus, which is covered by several markers, representing a fewer number of atoms. If all significant atoms in this region were visualized, at least 10 graphs for the same region would be produced. To prevent this, atoms have to be identified, which belong to the same set of CNVs. Accordingly, peaks are defined by the rule described above to reduce the number of regional overview plots.

```
do.log.regression(CNVdata = objCNVineta,
batch.case=c("YRI"), batch.control=c("CEU"), draw=TRUE, out.file="log.regression")
```

The region on chromosome 11 exemplifies the concept of "atoms" and "peaks". The chromosomal region marked by dotted lines consists of several atoms. Deletions and duplications can be seen at various positions in different samples. However, CNVineta identifies the entire region between both dotted lines to be associated.



In the above example, CEU samples are set as cases and YRI samples as controls. Significant differences between "cases" and "controls" were found for various regions, demonstrating that frequencies of CNVs in this regions varies significantly between various ethnicities. To reduce signals resulting from population stratification, covariates can be added to the logistic regression model. Within the example data set, an additional file with eigenvectors for each sample is provided. These eigenvectors were generated with plink [4]. In the following example, the first two of the four calculated eigenvectors were used as covariates.

```
do.log.regression(CNVdata = objCNVineta,
batch.case=c("YRI"), batch.control=c("CEU"), draw=FALSE,cov.header = TRUE,
cov.idx = 1, cov.columns = c("dim1","dim2"), cov.sep = "\t",
covariates = "CNVineta.HapMap.Affy6.0.eigenvectors.txt")
```

Adding these covariates to the model, the corrected p-values are close to 1. Accordingly, the

previously observed differences between CEU and YRI seem to be associated with the eigenvectors, which -in this case- are correlated with the geographic origin of the samples [5]. The covariates table can be added in different ways: as a separate file, as outlined above or as a separate data.frame.table. The parameter *cov.idx* provides the *do.log.regression* function with the column containing the sample ids, which can be a numeric index or a character depending on the column header name. The same applies for the *cov.columns* parameter, which provides information on columns containing the covariate that should be included in the analysis. For more details, use the help functionality (*?do.log.regression*) within the R package.
Keep in mind that it is necessary to clean your data/sample set before doing data mining.

Generating CNVineta input files

CNVineta is a post-processing and data mining tool and is capable of analyzing data from various different platforms once the data has been converted to the CNVineta input file format. Data conversion functionality is implemented in the CNVineta package and described later on in this tutorial. We start with a description of the CNVineta input file format. Users may choose to either use the implemented R functionality with all the advantages of generic R functions like *read.table* or convert data using custom-made scripts/executables for best runtime performance. For questions or special solutions please refer to http://groups.google.com/group/cnvineta.

CNVineta input file format description

In order to provide both a quick overview of the overall CNVs in case-control data sets as well as access to the LRR and BAF raw data of individual samples, CNVineta requires a specific file format. An example data set, which has already been converted to the CNVineta format, is available at the CNVineta website. CNVineta requires the following 5 files:

(1) The **SNP array annotation file** is needed to assign genomic coordinates to specific markers and to locate the raw data in the binary file (3). The tab-delimited file contains information on marker name, a zero-based numeric index for internal usage, as well as chromosome and physical start and end position. The SNP array annotation file is mandatory for allowing CNVineta to assign raw data locations in the binary file by using the zero-based numeric index. No header or comment lines are allowed and the column order from left to right is "marker id", "chromosome", "start position", "end position" and "zero based index".

```
SNP_A-4261452     2     7430093     7430093     251547
...
CN_922409    22    49581309    49581334    1880793
```

Marker names starting with "CN" will be identified as non-polymorphic probe sets, marker names starting with "SN" or "rs" as SNP probe sets. If your probe set annotation does not include start and end coordinates but position, use the same entry for start and end. The markers must be ordered by chromosome, followed by start and end coordinates. The order of chromosomes is irrelevant, but all markers of the same chromosome must be in one block and ordered by position. The zero-based index should be applied to the sorted marker table.

(2) The **sample annotation file** is a list of all samples included in the analysis. The tab-delimited file stores information on sample name, a zero-based numeric index for internal usage and the gender. The sample annotation file is mandatory for allowing CNVineta to assign positions in the binary file (3) by the zero based numeric index. No header or comment lines are allowed and the column order from left to right is "sample id", "zero based index" and "gender".

```
NA06985     0     female
...
NA12003     269   male
```

For the gender column only the entries "female","male" and "unknown" are allowed.

(3) The **binary.dat** file contains the raw data on LRR and BAF for all samples. Depending on the size of the sample set, the binary file can be several gigabytes in size. However, despite the size of the binary file, CNVineta was designed with quick access to raw data in mind, and data from case-control cohorts of thousands of individuals can be visualised rapidly from a standard desktop computer. For each marker of each sample two numeric values, LRR followed by BAF, are stored in binary format. The data type is a C++ float stored in little endian format. To access the LRR and BAF of a specific marker of a specific sample, the zero-based indices belonging to sample and marker and the number of markers on the SNP array have to be known. An example:

- Array with 10,000 marker
- We are interested in the SNP with the zero based index 10
- The sample of interest has the zero based index 14
- Keep in mind that a C++ float value allocates 4 byte (LRR + BAF 8 byte)

=> LRR = read 4 bytes at (10,000*14+10)*8 and convert to float
=> BAF = read 4 bytes at (10,000*14+10)*8+4 and convert to float

Accordingly, if the SNP array includes 1,000,000 markers and the store raw data of 10,000 samples, then the binary file must have a size of exactly 80,000,000,000 byte (~ 74.5 GB). The 4 bytes of the R specific "NA" value (missing value) in hexadecimal are translated to :

00 00 00 80

(4) The **segment annotation file** is a table of predicted CNVs in the sample set. For each CNV, information on the individual, batch, copy number state, number of marker covering the segment, average marker distance in kb, chromosome, start position, end position, start marker and end marker is stored. Several well-established algorithms are available for CNV prediction and CNVineta is not limited to a specific algorithm. The provided HapMap example data set has been analyzed with Affymetrix® Power Tools.

```
NA18582  CHB  1  0   5  3   31477964  31478724  SNP_A-4263432  SNP_A-2315022
...
NA18521  YRI  3  53  25 22  41227214  41280423  CN_913521      CN_913541
```

Header or comment lines are not allowed in this tab-delimited file and the column order from left to right is the same as the column order above, starting with individual and ending with end marker. Please pay close attention to sample and marker identifiers. The individual column has to contain the same sample identifiers as used in the sample annotation file (2). The start marker and end marker entries have to be annotated in the "marker id" column of the SNP array annotation file (1).

(5) The **RefGene file** is a standard list of all genes in the human genome as provided by the UCSC genome browser [3] (http://genome.ucsc.edu). Since annotations of the human genome are subject to change, CNVineta is not limited to a specific build of human genome (the HapMap example uses UCSC hg18). The RefGene file is provided separately rather then connecting to online database, allowing for CNVineta to function as a stand-alone application.

Converting CNV calling results to CNVineta input file format

For Illumina data, please follow the instructions provided in Supplementary Figure S3.
CNV calling tools usually provide at least two file types, which are necessary to generate CNVineta input files. These include a segment report, which lists all copy number variation that were called in the sample set (1) and a raw data file which includes the raw data (LRR and BAF) of all markers (2). Sometimes, the raw data file is already used as input for the CNV calling tool (e.g. QuantiSNP). With all other CNV calling algorithms, both file types have to be provided to make the data compatible for CNVineta. The array annotation file (3) is usually provided by the array manufacturer. If not provided, the required information can usually be derived from the raw data files (2), as well. These are input files required for CNVineta in detail:

(1) The **copy number segments file** stores information about all CNVs, which were called by the CNV calling algorithm. This file includes information about the chromosomal location of each CNV as well as start and end coordinates. The fourth required column is the copy number state.

(2) The **raw data file** stores the LRR and BAF values of every marker. To assign the raw data correctly, the required columns are "marker id", "LRR" and "BAF".

The **marker annotation file** is a list of all markers on the array and is required to have columns storing information on "marker id", "chromosome", "start" and "end". Sometimes, markers are not annotated with start- and end-coordinates but merely positions. In this case, when reading the annotation file into R, triggers can be set that allow the R function to use the position value for start and end positions. If an annotation file is not available, the raw data file (2) can be used instead. Usually, the raw data file also contains information on chromosome and chromosomal position, which is sufficient in combination with the marker id. However, be aware of inconsistency. A raw data file may only have a subset of marker ids which might vary between different raw data files.

Before continuing with an example conversion process we would like to point out the CNVineta website, where three conversion examples are available for download, including examples for Affymetrix® Power Tools CNV calls, QuantiSNP and a Java application to convert Illumina® Genomestudio output to the CNVineta format. Please visit http://www.ikmb.uni-kiel.de/cnvineta/ and download these examples from the conversion section.

In general, the Affymetrix® Power Tools and QuantiSNP output is very similar and these two examples are used to demonstrate how typical CNV calling results can be processed to generate CNVineta input files. Illumina® data is different, as it provided with a large data table, which contains raw data and CNV states for the entire sample set in a single file. To convert this to the CNVineta format, a Java application is provided for the conversion process. This Java application can be used for all Genomestudio plugins used for CNV analysis.

An example conversion process

The following commands need to be adapted. After loading the CNVineta library, create an object of class `"CNVinetaPreprocessor"`.

```
library(CNVineta)
objFileConverter = new("CNVinetaPreprocessor")
```

Set a parameter, which tells the functions if data should be added to an existing CNVineta input files or if new input files should be generated. The following command evokes the generation of new input files.

```
objFileConverter@add.data=FALSE
```

Now the CNVineta object is ready for loading the annotation data. If the *add.data* trigger is set to TRUE, the following commands can be skipped. Marker annotation can be added step by step. For example, Affymetrix® provides two separate annotation files for SNPs and non-polymorphic probe sets. The function *addMarkerAnnotation* internally uses the generic *read.table* command. All triggers available for *read.table* are also available for *addMarkerAnnotation*, allowing for customization of this command. Refer to *?addMarkerAnnotation* for more details. An example command for Affymetrix® SNP array 6.0 annotation version "na29" located at the folder "/foo/bar" could look like this:

```
objFileConverter =addMarkerAnnotation(dbCreator = objFileConverter,
                        annotation.SNP="/foo/bar/GenomeWideSNP_6.na29.annot.csv",
                        header=TRUE,
                        marker.column="Probe.Set.ID",
                        chrom.column="Chromosome",
                        pos.start="Physical.Position",
                        pos.end="Physical.Position",
                        sep = ",",
                        comment.char = "#",
                        na.strings = "---")

objFileConverter =addMarkerAnnotation(dbCreator = objFileConverter,
```

```
annotation.SNP="/foo/bar/GenomeWideSNP_6.cn.na29.annot.csv",
header=TRUE,
marker.column="Probe.Set.ID",
chrom.column="Chromosome",
pos.start="Chromosome.Start",
pos.end="Chromosome.Stop",
sep = ",",
comment.char = "#",
na.strings = "---")
```

As can be seen in the example above, the trigger *header=TRUE* is set. In this case, header names can be given for the required columns. If *header=FALSE*, column number instead of names have to be set. When using column header names you should keep in mind that R converts special characters to dots. That is the reason why *marker.column="Probe.Set.ID"* was used instead of *marker.column="Probe Set ID"*! In case of doubt, column numbers always work. As the next step, output file names have to be defined by setting the slot *out.file* of the *objFileConverter* object. This should be the full path to the binary file that should be created or extended. For example, if a binary file called "Test.binary" located at the folder "/foo/bar" should be created, use the following command:

```
objFileConverter@out.file="/foo/bar/Test.binary"
```

Please choose the filename careful. If pointed to an existing binary file with the trigger *add.data=FALSE*, all files will be deleted before file conversion starts. If data should be added to an existing file, set the trigger *add.data=TRUE*. In addtion, *objFileConverter @out.file* has to point to the binary file including all other CNVineta input files in the same folder. The final step before executing *createCNVinetaDb* is setting up a table, which provides information on the location of segment files (1) and raw data files (2) and includes information on sample ID, gender and batch. This table should have the following column names: "raw.data.file", "segment.file", "sample.name", "sample.gender" and "batch". Column entries should contain the corresponding information. A tab-delimited example file called "example.txt" located at "/foo/bar" could look like this:

```
raw.data.file   segment.file     sample.name  sample.gender  batch
/data/raw/1.dat /data/CNV/1.txt  Sample1      female         ShipmentA
...
/data/raw/N.dat /data/CNV/N.txt  AnotherName  male           ShipmentB
```

The following command will load that file to the slot *the.data.table* of the *objFileConverter* object.

```
objFileConverter@the.data.table = read.table("/foo/bar/example.txt", sep="\t", header=TRUE)
```

Now everything is ready to start the data conversion with the command *createCNVinetaDb*. For a detailed description of the various triggers for that command please type ?*createCNVinetaDb* in your R command prompt. An example command, which uses the most important triggers, could look like this:

```
createCNVinetaDb(dbCreator =objFileConverter,
                 raw.marker.column="Probe.Set",
                 raw.LRR.column="Log.R.Ratio",
                 raw.BAF.column="B.allele.Frequency",
                 seg.chrom.column="Chromosome",
```

seg.start.column="Start..bp.",
seg.end.column="End..bp.",
seg.state.column="Copy.Number",
raw.header=TRUE,raw.sep = "\t",
seg.header=TRUE,seg.sep = "\t")

For both input file types, i.e. segment files (1) and raw data files (2), all triggers available for the generic R command *read.table* are also available via *createCNVinetaDb*. This allows customization of the specific input file format. For segment files, the prefix "seg." and for the raw data files the prefix "raw." should be used as indicated by the arguments "*raw.header*" and"*seg.header*".References

1. International HapMap Consortium, The International HapMap Project *Nature*, 2003. 426(6968): 789-96
2. McCarroll S.A. et al. (2008) Deletion polymorphism upstream of IRGM associated with altered IRGM expression and Crohn's disease. *Nat Genet.*, 40(9):1107-12
3. Karolchik D, et al. (2009) The UCSC Genome Browser. *Curr Protoc Bioinformatics.* Dec;Chapter 1:Unit1.4.
4. Purcell S et al. (2007) PLINK: a toolset for whole-genome association and population-based linkage analysis. *American Journal of Human Genetics*, 81.
5. Lao O. et al. (2008) Correlation between genetic and geographic structure in Europe. *Curr Biol.* 18(16):1241-8.
6. Zhang D. et al. (2008) Singleton deletions throughout the genome increase risk of bipolar disorder. Mol Psychiatry. 14(4):376-0